正课:

1. 添加/删除

2. HTML DOM常用对象

3. 什么是BOM  
4. window

5. 打开和关闭窗口

6. history

7. location

8. navigator

1. 添加/删除:

添加: 3步:

1. 先创建一个空元素对象:

var a=document.createElement("a");

相当于: <a></a>

2. 设置关键属性:

a.innerHTML="go to tmooc"; => <a> go to tmooc </a>

a.href="http://tmooc.cn"

=> <a href="http://tmooc.cn"> go to tmooc </a>

3. 将新元素添加到DOM树上指定位置

3种:

1. 父元素.appendChild(a) 将a追加到父元素下最后一个孩子之后

2. 父元素.insertBefore(a, child) 将a插入在父元素下一个现有child元素之前

3. 父元素.replaceChild(a, child) 用a代替父元素下现有的child子元素

优化: 尽量减少操作DOM树的次数:

为什么: 每修改一次DOM树，就会导致重排重绘，降低页面加载的效率。

原理:

html->DOM Tree ←修改

↓

Render Tree -> 重排->重绘paint

↑

css -> CSS Model

解决: 2种:

1. 当同时添加父元素和子元素时，应该先在内存中将所有字元素加入父元素下，最后再一次性将父元素添加到DOM树

2. 如果父元素已经在页面上了，要同时添加多个平级子元素时，应该找一个临时的父元素，先将所有子元素加入临时的父元素中，最后，再将临时的父元素添加到DOM树

临时的父元素: 文档片段 document fragment

什么是: 内存中，临时存储多个子元素的虚拟父元素

何时: 要同时添加多个平级子元素到DOM树时

如何: 3步:

1. 创建文档片段对象:

var frag=document.createDocumentFragment();

2. 将子元素添加到文档片段中:

frag.appendChild(child)

3. 将frag整体一次性添加到DOM树

父元素.appendChild(frag)

删除: 父元素.removeChild(child)

child.parentNode.removeChild(child)

2. HTML DOM 常用对象:

Image: 唯一的简化: var img=new Image();

Select/Option

Select:

属性: .selectedIndex 自动获得当前选中的option的下标i

.options 快速获得当前select下所有option的集合

.options.length 表示select下option的个数

.length 简化.options.length

方法: .add(opt) 简化.appendChild(opt)

问题: 不支持文档片段

.remove(i) 移除i位置的一个option

Option:

最大的简化: var opt=new Option(text, value)

新建一个option元素，同时设置其内容和值

Table/...:

逐级管理:

table管着行分组:

创建(行分组): 创建行分组，同时将行分组立刻就添加到table中

var thead=table.createTHead()

//createElement(thead);table.appendChild(thead)

var tfoot=table.createTFoot()

var tbody=table.createTBody()

删除(行分组):

table.deleteTHead();

table.deleteTFoot();

获得(行分组):

table.tHead

table.tFoot

table.tBodies[i]

行分组管着行:

创建一个新行:

var tr=行分组.insertRow(i);

在行分组的第i行位置前插入一个新行，并返回新行对象。原i位置的行及其之后的行被向后挤压顺移一位

固定套路:

1. 在行分组末尾追加一个新行:

var tr=行分组.insertRow()

//createElement(tr) 行分组.appendChild(tr)

2. 在行分组第一行之前插入一个新行:

var tr=行分组.insertRow(0)

删除现有行:

不好: 行分组.deleteRow(i)

问题: 因为.前的主语是行分组，所以i要求是行在行分组内的相对下标位置。而tr有一个rowIndex属性，虽然可获得行的下标位置，但是是相对于整个表格的位置。

好的: table.deleteRow(tr.rowIndex)

获取某一个行: 行分组.rows[i]

行管着格:

添加: var td=tr.insertCell(i) //省略i，默认在末尾追加

局限: 只能添加td，不能添加th

删除: tr.deleteCell(i)

获取: tr.cells[i]

Form/表单元素